Linear Optimization over a Polymatroid with Side Constraints – Scheduling Queues and Minimizing Submodular Functions

Yingdong Lu
IBM T.J. Watson Research Center
Yorktown Heights, N.Y.10598

David D. Yao
IEOR Dept, 302 Mudd Building
Columbia University, New York, NY 10027

April 2007

Abstract

Two seemingly unrelated problems, scheduling a multiclass queueing system and minimizing a submodular function, share a rather deep connection via the polymatroid that is characterized by a submodular set function on the one hand and represents the performance polytope of the queueing system on the other hand. We first develop what we call a *grouping* algorithm that solves the queueing scheduling problem under side constraints, with a computational effort of $O(n^3LP(n))$, n being the number of job classes, and $LP(n)$ being the computational efforts of solving a linear program with no more than n variables and n constraints. The algorithm organizes the job classes into groups, and identifies the optimal policy to be a priority rule across the groups and a randomized rule within each group (to enforce the side constraints). We then apply the grouping algorithm to the submodular function minimization, mapping the latter to a queueing scheduling problem with side constraints. We show the minimizing subset can be identified by applying the grouping algorithm n times. Hence, this results in a algorithm that minimizes a submodular function with an effort of $O(n^4LP(n))$.

1 Introduction

Linear optimization over a polymatroid — a polytope that possesses matroid properties — is known to be solvable through a greedy algorithm; refer to Edmonds [10], Dunstan and Welsh [9], and Welsh [32]. It is also known that the performance space of a multiclass queueing system is a polymatroid, provided the system satisfies strong conservation laws; refer to Shanthikumar and Yao [30], and also the earlier works of Coffman and Mitrani [6], and Federgruen and Groenevelt [12, 13]. With this connection, the control or dynamic scheduling of multiclass queues can be translated into an optimization problem; and under a linear objective, the optimal control

is readily identified as a priority rule associated with one of the vertices of the performance polytope.

In many scheduling applications, there are additional, "side constraints" on the performance; for instance, the delay of jobs should not exceed a certain limit, or the completion rate (throughput) of jobs must meet a certain requirement. The dynamic scheduling of multiclass queues under such side constraints is still polynomially solvable, as argued in Yao and Zhang [33], directly applying a result by Grötschel, Lovász and Shrijver [19]. This argument, however, is essentially an "existence proof." It does not explicitly identify an optimal policy, let alone one with a simple structure that appeals to implementation.

In this paper, we develop a combinatorial algorithm that identifies the optimal policy for scheduling multiclass queues with side constraints. This is what we call a "grouping algorithm," which partitions the n job classes into groups, and the associated optimal policy is a priority rule across the groups, while enforcing the side constraints (via randomization, for instance) among the classes within each group. The grouping algorithm identifies the optimal policy with a computational effort of $O(n^3 \times LP(n))$, where $LP(n)$ refers to the complexity of solving a linear program with number of variables and constraints bounded by n.

It turns out that this grouping algorithm can also be used to solve the problem of minimizing a submodular function — finding the subset that minimizes a set function, $\psi : 2^E \mapsto \mathcal{R}$, which maps the subsets of a given set E to the real line and satisfies submodularity (refer to Definition 1 (iii)). Here the key ingredient is that a certain representation of the submodular function minimization can be turned into a queueing scheduling problem with side constraints, with the submodular function relating to the polymatroid that is the performance polytope of the queueing system. We show that the minimizing subset can be identified by applying the grouping algorithm $n = |E|$ times; each time, it identifies a subset, and only these subsets are candidates for optimality. Hence, the overall computational effort is $O(n^4 LP(n))$.

Submodular function minimization (SFM), widely regarded as "the most important problem related to submodular function" ([20]), has attracted much research effort in recent years. Here we briefly highlight the relevant literature, as the details are now available in two recent survey papers by Fleischer [14] and McCormick [27].

Using the ellipsoid approach, Grötschel, Lovász and Shrijver [19] established that the SFM is polynomially solvable. However, the ellipsoid technique is not usually a practical algorithm, and it does not generate much combinatorial insight. Cunningham [7] proposed a max-flow type of approach based on linear programming (LP), which results in a combinatorial algorithm that

solves the SFM in pseudo-polynomial time. (The qualifier "pseudo" refers to the fact that the running time is not only a polynomial in n, but also a polynomial in M, an upper bound of the set function.) Recently, combinatorial algorithms that solves SFM in strongly polynomial time (meaning, the running time is a polynomial in n only, and is independent of M) have been independently developed by Iwata, Fleicher and Fujishige [23], and by Shrijver [31]; and further improvement has appeared in Iwata [22, 21], and Fleischer and Iwata [15]. The starting point of these algorithms is an equivalent problem over a polymatroid with side constraints, and this LP is solved via an augmenting-path approach. In contrast, our approach to the SFM is to solve a sequence of n LP problems; each is an LP over a polymatroid with side constraints, but the inequalities in the side constraints are in the opposite direction to those in existing SFM algorithms. None of these n LP's is equivalent to the SFM, but solving all of them does yield the solution to the SFM. More importantly, each such LP can be represented as a queueing scheduling problem highlighted above, and hence can be solved by the grouping algorithm. (Refer to more details in §5, Remarks 1 and 2 in particular.) The advantage of our algorithm is that it is fully combinatorial (meaning, it only uses addition, subtraction and comparison); and its running time of $O(n^4 LP(n))$ matches that of the strongly polynomial version of the ellipsoid algorithm, and is significantly better than the fastest existing combinatorial algorithms. (For instance, the one due to Iwata [21] has a running time of $O(n^7)$; refer to [27].)

Briefly, the rest of the paper is organized as follows. In the next section, we review preliminary materials for linear optimization over a polymatroid, and its connection to ququeing scheduling via strong conservation laws. In §3 we collect the essentials of the grouping algorithm that solves the queueing scheduling problem with side constraints, in particular the conditions that will ensure its feasibility and optimality. This is folllowed by a detailed description of the algorithm in §4, where we also show that the required conditions outlined in §3 are all met. In §5 we apply the grouping algorithm to the SMF problem. Finally, in §**??** we consider a special case when the submodular function satisfies the so-called generalized symmetry.

The polymatroid connection to queueing scheduling has in recent years been extended to a more general framework via so-called generalized conservation laws; refer to the papers by Bertsimas and his co-workers, e.g., [1, 2, 3]; also refer to Lu [26], Zhang [34], and Chapter 11 of Chen and Yao [8]. In this paper, however, we choose to stay within the more restrictive framework of polymatroid, since our emphasis here is not on queueing scheduling per se, but rather on the interplay between such scheduling problems under side constraints and minimizing submodular functions.

2 Preliminaries

2.1 LP over a Polymatroid

Throughout, $E = \{1, \ldots, n\}$ is a finite set; $x = (x_i)_{i \in E}$ is a vector, and $x(A) := \sum_{i \in A} x_i$ for any $A \subseteq E$. We shall use "increasing" and "decreasing" in the non-strict sense, to mean non-decreasing and non-increasing, respectively.

Definition 1 (Welsh [32]) Given $E = \{1, 2, ..., n\}$, the following polytope,

$$\mathcal{P}(f) = \{x \geq 0 : x(A) \leq f(A), A \subseteq E\}, \tag{1}$$

is called a polymatroid if the function $f : 2^E \to \mathcal{R}_+$ satisfies the following properties:
(i) (*normalized*) $f(\emptyset) = 0$;
(ii) (*increasing*) if $A \subseteq B \subseteq E$, then $f(A) \leq f(B)$;
(iii) (*submodular*) if $A, B \subseteq E$, then $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$

Let π be a permutation of $\{1, 2, ..., n\}$, and define a vector x^π with the following components:

$$\begin{aligned} x^\pi_{\pi_1} &= f(\{\pi_1\}), \\ x^\pi_{\pi_2} &= f(\{\pi_1, \pi_2\}) - f(\{\pi_1\}), \\ &\cdots \\ &\cdots \\ x^\pi_{\pi_n} &= f(\{\pi_1, \pi_2, ..., \pi_n\}) - f(\{\pi_1, \pi_2, ..., \pi_{n-1}\}). \end{aligned} \tag{2}$$

Then, we have another definition for polymatroid:

Definition 2 The polytope $\mathcal{P}(f)$ in (1) is a polymatroid if $x^\pi \in \mathcal{P}(f)$ for any permutation π.

That the two definitions, 1 and 2, are equivalent is a classical result; refer to, e.g., Edmonds [10], Welsh [32], and Dunstan and Welsh [9]. The following theorem, also a classical result, can be found in these references as well.

Theorem 1 ([9, 10, 32]) Consider the LP problem: $\max_{x \in \mathcal{P}(f)} \sum_{i \in E} c_i x_i$. Suppose $\pi = (\pi_i)_{i \in E}$ is a permutation, such that

$$c_{\pi_1} \geq c_{\pi_2} \geq \cdots \geq c_{\pi_n}.$$

Then, x^π as specified in (2) is the optimal solution.

For the polymatroid $\mathcal{P}(f)$, we call the following polytope its *base*:

$$\mathcal{B}(f) = \{x \geq 0 : \, x(A) \leq f(A), \, A \subset E; \, x(E) = f(E)\}. \tag{3}$$

In queueing applications, it is useful to have another type of base, in addition to the one in (3):

$$\mathcal{B}(b) := \{x \geq 0 : \, x(A) \geq b(A), \, A \subset E; \, x(E) = b(E)\}, \tag{4}$$

where $b(\cdot)$ is increasing and supermodular, with $b(\emptyset) = 0$. It is straightforward to verify that $\mathcal{B}(b) = \mathcal{B}(f)$ by letting $f(A) = b(E) - b(E - A)$. Since the resulting f is submodular, increasing, and $f(\emptyset) = 0$, we know $\mathcal{B}(b)$ is also the base of a polymatroid. In particular, each of its vertices can be expressed as x^π as in (2), with $f(\cdot)$ replaced by $b(\cdot)$.

2.2 Polymatroid Structure in Multiclass Queues

Let $E = \{1, 2, ..., n\}$ denote the set of all job classes (or, types) in a queueing system. Let $x = (x_i)_{i \in E}$ denote the performance vector, e.g., waiting time (delay) or service completions (throughput). Let $\mathcal{U}$ denote the family of all *admissible* policies that control the way jobs are served in the system. This includes all policies that are non-anticipative (i.e., no clairvoyance) and non-idling (i.e., serve whenever there is a job to be served). We have:

Definition 3 (Shanthikumar and Yao [30]) A performance vector x of a stochastic system with a set of multiple job classes, $E = \{1, ..., n\}$, is said to satisfy conservation laws, if there exists a set function b (resp. f): $2^E \mapsto \mathcal{R}_+$, satisfying

$$b(A) = \sum_{i \in A} x_{\pi_i}, \quad \forall \pi : \{\pi_1, ..., \pi_{|A|}\} = A, \quad \forall A \subseteq E; \tag{5}$$

or respectively,

$$f(A) = \sum_{i \in A} x_{\pi_i}, \quad \forall \pi : \{\pi_1, ..., \pi_{|A|}\} = A, \quad \forall A \subseteq E; \tag{6}$$

such that under any admissible control, $u \in \mathcal{U}$, the following is satisfied:

$$\sum_{i \in A} x^u_{\pi_i} \geq b(A), \quad \forall A \subset E; \qquad \sum_{i \in E} x^u_{\pi_i} = b(E);$$

or respectively,

$$\sum_{i \in A} x^u_{\pi_i} \leq f(A), \quad \forall A \subset E; \qquad \sum_{i \in E} x^u_{\pi_i} = f(E).$$

Intuitively, the above definition says:

- The total performance over all job classes in E is invariant: it is a constant that is policy independent.
- The total performance over any given subset, $A \subset E$, of job types is minimized (e.g., if x is delay) or maximized (e.g., if x is throughput) by offering priority to job types in A over all other types in $E - A$.
- The function $b(A)$ or $f(A)$ is the corresponding performance when A is given priority over $E - A$. Furthermore, the permutation π represents a priority rule such that π_1 is given the highest priority; π_2, the second highest priority; and so forth. And the requirement, $\forall \pi : \{\pi_1, ..., \pi_{|A|}\} = A$, signifies that $b(A)$ and $f(A)$ depend only on the composition of the job types in A; they are independent of the priorities among these job types.

The above definition of conservation laws, along with Definition 2, immediately leads to the important fact that when conservation laws are satisfied, the performance space over all admissible controls is a polymatroid — in fact, the base polytope of a polymatroid, and the right-hand-side function is determined by the conservation laws too, i.e. the performance resulting from assigning priority to certain subset of jobs. This is simply because any π, a priority rule, is admissible, and hence, x^π belongs to the performance polytope, and x^π following (5) and (6) are of the same form as x^π in (2).

Suppose we want to find the optimal control — a dynamic scheduling rule — of the n job types in the set E, so as to maximize an objective function with respect to a performance vector x. The problem can be expressed as follows:

$$\max_{u \in \mathcal{U}} \sum_{i \in E} c_i x_i^u, \tag{7}$$

with c_i's being the costs. Suppose x satisfies conservation laws; and hence, the performance space under all admissible controls is the base polytope of a polymatroid, denoted $\mathcal{B}(f)$. So we can first solve a linear programming problem:

$$\max_{x \in \mathcal{B}(f)} \sum_{i \in E} c_i x_i.$$

From Theorem 1, we know the optimal solution is x^{π^*}, where π^* is a permutation that orders the cost coefficients in decreasing order:

$$c_{\pi_1^*} \geq c_{\pi_2^*} \geq \cdots \geq c_{\pi_n^*}.$$

We can then identify a control u^* that realizes this performance. The fact that the solution being a vertex of the performance space ensures that the optimal policy is a priority rule. It is clear that to derive the optimal policy, there is no need to compute the right-hand-side function of the performance polytope. For more details regarding conservation laws and the connection to polymatroid, including many examples, refer to Chapter 11 of Chen and Yao [8].

3 Side Constraints: Feasibility and Optimality

Consider the polymatriod $\mathcal{P}(f)$ in (1). Suppose there are additional constraints: $x_i \geq d_i \geq 0$, $i \in E$. (For instance, d_i is the requirement on the throughput of type i jobs.) Adding these to $\mathcal{P}(f)$ results in the following polytope:

$$\mathcal{P}'(f) = \{x(A) \leq f(A), A \subseteq E;\, x_i \geq d_i, i \in E\} \tag{8}$$

We want to solve the following LP problem:

$$\max_{x \in \mathcal{P}'(f)} \sum_{k=1}^{n} c_k x_k. \tag{9}$$

For the LP in (9) to be feasible, the following condition is readily verified.

Lemma 2 (Ross and Yao [28]) The polytope $\mathcal{P}'(f)$ is non-empty if and only if $d := (d_1, ..., d_n) \in \mathcal{P}(f)$.

Below, we shall assume that $d \in \mathcal{P}(f)$ always holds. A direct application of the following result establishes that the LP in (9) is polynomially solvable.

Theorem 3 (Grötschel, Lovász, and Schrijver [19]) Let $\mathcal{K}$ be a class of convex bodies. There exists a polynomial algorithm to solve the separation problem for members of $\mathcal{K}$, if and only if there exists a polynomial algorithm to solve the optimization problem for the members of $\mathcal{K}$.

Making use of this result, we know that the membership test problem (which is equivalent to the separation problem) over $\mathcal{P}(f)$ is polynomially solvable, since the LP over $\mathcal{P}(f)$ is. Hence, the membership test over $\mathcal{P}'(f)$ is also polynomially solvable, since this amounts to checking the additional n side constraints. Therefore, the LP over $\mathcal{P}'(f)$ is also polynomially solvable.

However, the argument in [19] is based on an ellipsoid approach. In contrast, below we develop a fully combinatorial algorithm that solves the LP over $\mathcal{P}'(f)$ in $O(n^4)$ steps, and also reveals the structure of the optimal solution.

Without loss of generality, assume the coefficients in (9) are ordered as follows:

$$c_1 \geq c_2 \geq \cdots \geq c_n > 0. \tag{10}$$

The algorithm below divides the n variables into $m \leq n$ groups: $G_1, ..., G_m$. Denote, $G^{(0)} = \emptyset$; and

$$G^{(i)} := G_1 \cup G_2 \cup \cdots \cup G_i, \qquad i = 1, ..., m.$$

Within each group G_i, there is exactly one "lead" variable, indexed ℓ_i, which takes on a value $x_{\ell_i} \geq d_{\ell_i}$; the rest of the group, denoted G_i^S (possibly empty), consists of variables that all take values *at* the constraints, i.e.,

$$x_{k_i} = d_{k_i}, \quad k_i \in G_i^S, \quad i = 1, ..., m. \tag{11}$$

Furthermore, the lead variable takes the following value:

$$x_{\ell_i} \;\; = \;\; f(G^{(i)}) - f(G^{(i-1)}) - d(G_i^S), \quad i = 1, ..., m; \tag{12}$$

where we have used the notation $d(A) := \sum_{i \in A} d_i$, for $A = G_i^S$.

Lemma 4 The solution x as specified in (11) and (12) is feasible, i.e., $x \in \mathcal{P}'(f)$, if for each $i = 1, ..., m$, we have: (a) $x_{\ell_i} \geq d_{\ell_i}$, and (b)

$$x(A) \leq f(G^{(i-1)}; A) - f(G^{(i-1)}), \quad A \subset G_i. \tag{13}$$

When the above do hold, we have

$$x(G^{(i-1)}; A) \leq f(G^{(i-1)}; A), \quad A \subset G_i; \qquad x(G^{(i)}) \;\; = \;\; f(G^{(i)}); \tag{14}$$

for all $i = 1, ..., m$.

Proof. First, all side constraints are satisfied, given (a), along with (11). Also note that from (12), since $x(G_i) = x_{\ell_i} + d(G_i^S)$, we have

$$x(G_i) = f(G^{(i)}) - f(G^{(i-1)}),$$

which, along with (13), leads to the relations in (14).

Next, we show $x(A) \leq f(A)$, for any $A \subseteq E$. Write $A = \cup_{i=1}^m D_i$, with $D_i \subseteq G_i$, $i = 1, ..., m$. Similar to the $G^{(i)}$ notation, denote $D^{(i)} := \cup_{k=1}^i D_k$, and $D^{(0)} := \emptyset$. Then, from (13), we have

$$\begin{aligned} x(D_i) \;\; &\leq \;\; f(G^{(i-1)}, D_i) - f(G^{(i-1)}) \\ &\leq \;\; f(D^{(i)}) - f(D^{(i-1)}) \end{aligned}$$

where the second inequality follows from submodularity. Hence, summing over all i, taking into account that D_i's are non-overlapping (since the G_i's are), we have

$$x(A) = \sum_{i=1}^{m} x(D_i) \leq f(D^{(m)}) = f(A).$$

□

Theorem 5 The feasible solution characterized in Lemma 4 is optimal if the indices of the lead and nonlead variables satisfy the following relations:

$$\ell_j \leq \ell_{j+1},\ j = 1, ..., m-1; \qquad \ell_j \leq k_j, \quad k_j \in G_j^S,\ j = 1, ..., m. \tag{15}$$

Proof. The dual of the LP in (9) takes the following form:

$$\begin{aligned} \min \quad & \sum_{A \subseteq E} y_A f(A) - \sum_{i \in E} d_i z_i \\ \text{s.t.} \quad & \sum_{A \ni i} y_A - z_i \geq c_i, \qquad i \in E; \\ & y_A \geq 0, \quad z_i \geq 0; \qquad A \subseteq E,\ i \in E; \end{aligned} \tag{16}$$

where y_A and z_i are the dual variables (the latter corresponds to the side constraints).

Following (11) and (12), the primal objective value is:

$$\sum_{j=1}^{m} (c_{\ell_j} - c_{\ell_{j+1}}) f(G^{(j)}) - \sum_{j=1}^{m} \sum_{k_j \in G_j^S} (c_{\ell_j} - c_{k_j}) d_{k_j}, \tag{17}$$

where $c_{\ell_{m+1}} := 0$.

Now, let the dual variables take the following values:

$$y_A = c_{\ell_j} - c_{\ell_{j+1}}, \qquad A = G^{(j)}, \quad j = 1, ..., m;$$

let all other $y_A = 0$; let

$$z_{k_j} = c_{\ell_j} - c_{k_j}, \qquad k_j \in G_j^S, \quad j = 1, ..., m;$$

and let all other $z_i = 0$. Then, the non-negativity of these dual variables follows from (15), taking into account (10). Furthermore, consider the h-th constraint in the dual problem. Suppose $h \in G_j$. If h is a lead variable, i.e., $h = \ell_j$, then $h \in G^{(i)}$ for all $i \geq j$. Hence the constraint becomes:

$$\sum_{i \geq j} (c_{\ell_i} - c_{\ell_{i+1}}) = c_{\ell_j} = c_h.$$

If h is a non-lead variable, i.e., $h = k_j \in G_j^S$, then we have

$$z_h = z_{k_j} = c_{\ell_j} - c_{k_j} = c_{\ell_j} - c_h,$$

as well as $h \in G^{(i)}$ for all $i \geq j$. In this case, the constraint becomes:

$$\sum_{i \geq j} (c_{\ell_i} - c_{\ell_{i+1}}) - (c_{\ell_j} - c_h) = c_h.$$

The complementary slackness condition is also readily checked. (In particular, the dual objective is equal to the primal objective in (17).)

Finally, primal feasibility has already been established in Lemma 4. Hence, the optimality of both primal and dual solutions. □

4 The Grouping Algorithm

The grouping algorithm to be presented below forms the groups to maintain feasibility. In particular it maintains the distinction of non-lead variables and lead variables in (11) and (12), and enforces the two conditions in Lemma 4 — that the lead variable of each group satisfies the side constraint, and that (13) holds for each group. Furthermore, we introduce a grouping structure[1] within each group to form a hierarchical structure that is crucial for the feasibility proof, as well as track the order of variables entering the group. Optimality follows, as argued in Lemma 4 and Theorem 5 above.

Although not essential to understanding the algorithm, it would help intuition to view the groups as associated with different priorities, with G_1 having the highest priority, and G_m the lowest priority. The variables can be viewed as throughput associated with each class, whereas the side constraints represent the minimal requirements on the throughput. The lead variable in each group contributes to maximizing the objective function, a weighted sum of the throughput of all job classes. The non-lead variables are non-contributors; as such, they are moved into a higher-priority group parsimoniously just to satisfy their respective side constraints.

Inductively, assume we know how to do the grouping when $|E| = n-1$. Specifically, suppose this results in $m-1$ groups, $G_1, ..., G_{m-1}$. Within each group G_i, we have,

- lead variable ℓ_i;
- subgroup g_j^i, $j = 1, \cdots, j_i$;

[1]Lack a better choice, we name them subgroups. However, there is no connections between our problem and the algebra entities of group and subgroup

and they satisfies, $\forall k_j \in g_j^i$,

$$f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i) + d_{k_j} \leq f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, k_j) \tag{18}$$

and there exists at least one $k_\ell \in g_j^i$ such that,

$$\begin{aligned} &f(G^{(i-1)}, g_1^i, \cdots, g_{j-1}^i, k_j, k_\ell) - f(G^{(i-1)}, g_1^i, \cdots, g_{j-1}^i, k_j) \\ &+f(G^{(i-1)}; g_1^i, \cdots, d_{j-1}^i) + g_{k_j} < d_{k_\ell} \end{aligned} \tag{19}$$

and,

$$f(G^{(i)}) - f(G^{(i-1)}) - d(G_i^S) \geq d_{\ell_i}. \tag{20}$$

Now consider the added n-th dimension; in particular, we want to find where to put the new element n and what value to assign to x_n. In the algorithm presented below, this is accomplished in Step 1. Then, we need to adjust the newly generated group and the lower priority groups that are affected (by the new group) so as to maintain feasibility; and this is done in Steps 2 and 3 of the algorithm.

To start with, we let n itself form the m-th group.

The Grouping Algorithm

Step 1. (*Group Identification*)

- Find the largest $i \leq m$, such that

$$f(G^{(i-1)}) + d_n \leq f(G^{(i-1)}; n), \tag{21}$$

and

$$f(G^{(i)}) + d_n > f(G^{(i)}, n). \tag{22}$$

Step 2. (*Subgroup Identification*) Within group G^i, find the largest j, such that

$$f(G^{(i-1)}, g_1^i, \cdots, g_{j-1}^i) + d_n \leq f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, n), \tag{23}$$

form new subgroups according to the validity of the following inequality,

$$f(G^{(i-1)}, g_1^i, \cdots, g_{j-1}^i, n) + d_{k_j} \leq f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, n, k_j), \tag{24}$$

- if for all $k_j \in g^i_j$, (24) holds, then, n will becomes new subgroup g^i_j, and the indices for all the previous subgroups that have index larger and equal to j will have to increase by one;
- otherwise n will join the subgroup g^i_j.

Step 3. (*Subgroup Validation*) This step is to determine if there is a feasible solution that the variables in subgroup g^i_j are valued at the side constraints. This is equivalent to solve the membership problem for vector $(d_{k_1}, d_{k_2}, \cdots, d_{k_\ell})$ to the polymatroid defined by the following submodular function,

$$g(A) := f(G^{(i-1)}, g^i_1, \cdots, g^i_{j-1}, A) - f(G^{(i-1)}; g^i_1, \cdots, d^i_{j-1}) \tag{25}$$

i.e. the subgroup is valid if the following LP has feasible solution,

$$\min x_{k_1}, s.t. \mathbf{x} \in P(g), x_{k_i} \geq d_{k_i} \tag{26}$$

If the LP in (26) is founded to be infeasible, then this subgroup is invalid, hence to be dissolved. Then if $j > 1$, then we need to go back to Step 2 to identify the subgroup for all the member previously in g^i_j; if $j = 1$, the group G^i need to be dissolved, go back to Step 1 for group identification for each member previously in G^i. **Step 4.**

- Conduct validation step for subgroup g^i_{j+1}, g^i_{j+2};
- Repeat the procedure to all the variables that has been bypassed;

Theorem 6 The grouping algorithm generates a solution that follows (11) and (12), and satisfies both feasibility requirements (a) and (b) of Lemma 4, and the inequalities in (15) of Theorem 5. Hence, it is an optimal solution to the LP in (9).

Proof. We only need to check the feasibility. To do this, we need to check if our output satisfies equation (13). For any $A \subset G_i$, let us consider two cases, (i) $\ell_i \in G_i - A$; (ii) $\ell_i \in A$.
Case I: Certainly, we have, $A = \cup_{j=1}^{j_i} A_j$, with $A_j = A \cap g^i_j$. From the validation step, we know that for each j,

$$x(A_j) = d(A_j) \quad \leq \quad f(G^{(i-1)}, g^i_1, \cdots, g^i_{j-1}, A_j) - f(G^{(i-1)}; g^i_1, \cdots, g^i_{j-1})$$

Due to submodularity, we have,

$$x(A_j) = d(A_j) \quad \leq \quad f(G^{(i-1)}, A_1, \cdots, A_{j-1}, A_j) - f(G^{(i-1)}; A_1, \cdots, A_{j-1})$$

It is true for for each j, then, sum them up, it implies that (13) is true.

Case II:

For any subgroup g_j^i, let us denote $k_1^j, k_2^j, \cdots, k_h^j$ of its member according to the order they entered the subgroup. Hence, the inequality (22) must be satisfied in the following form,

$$\begin{aligned}
d_{k_1^j} &> f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i, k_1^j) - f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i), \\
k_{g_2^j} &> f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i, k_1^j, g_2^i) - f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i, k_1^j), \\
&\cdots\cdots \\
d_{k_h^j} &> f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i, k_1^j, \ldots k_{h-1}^j, k_h^j) - f(G^{(i-1)}; \ell_i, g_1^i, \ldots g_{h-1}^i);
\end{aligned}$$

Because when k_1^j enters the group, $G(i)$ is made of $G^{(i-1)}$, subgroups $g_1^i, \cdots, g_{j-1}^i$ and its lead member ℓ_i, so the inequality (22) takes the form of the first inequality, and so on.

Meanwhile,

$$\begin{aligned}
&f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i, k_1^j) - f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i), \\
&f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i, k_1^j, k_2^i) - f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i, k_1^j), \\
&\cdots\cdots \\
&f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; \ell_i, k_1^j, \ldots k_{h-1}^j, k_h^j) - f(G^{(i-1)}; \ell_i, g_1^i, \ldots g_{h-1}^i);
\end{aligned}$$

forms a solution to another polymatroid, from its property, we can conclude that,

$$d(A_j) > f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, g_j^i; \ell_i) - f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i; A_j^c, \ell_i)$$

where $A^c := G_i^S - A$ and $A_j^c = A^c \cap g_j^i$.

Therefore, for any $A \subseteq G_{i+1}^S$, we have

$$d(A) > f(G^{(i)}; A, \ell_{i+1}) - f(G^{(i)}; \ell_{i+1}). \tag{27}$$

$$\begin{aligned}
& x_{\ell_{i+1}} + d(A) \\
= \;& f(G^{(i)}; G_{i+1}) - f(G^{(i)}) - d(A^c) \\
\leq \;& f(G^{(i)}; G_{i+1}) - f(G^{(i)}) - [f(G^{(i)}; A^c, \ell_{i+1}) - f(G^{(i)}; \ell_{i+1})] \\
\leq \;& f(G^{(i)}; A, \ell_{i+1}) - f(G^{(i)}),
\end{aligned}$$

where the first inequality follows from (27), applied to A^c, and the second one follows from submodularity, i.e.,

$$f(G^{(i)}; G_{i+1}) - f(G^{(i)}; A^c, \ell_{i+1}) \leq f(G^{(i)}; A, \ell_{i+1}) - f(G^{(i)}; \ell_{i+1});$$

hence, the feasibility.

□

Consider the b-version of the performance polytope (in Definition 3) with side constraints:

$$\mathcal{P}'(b) = \{x(A) \geq b(A), A \subseteq E;\ x_i \leq d_i, i \in E\}. \tag{28}$$

(For instance, d_i is the upper limit on the delay of type i jobs.) We want to solve the following LP problem:

$$\min_{x \in \mathcal{P}'(b)} \sum_{i=1}^{n} c_i x_i. \tag{29}$$

Direct verification will establish that the Grouping Algorithm is readily adapted to solve the above problem: simply change f to b, and reverse the directions of all the inequalities involved. (Note that the side constraints are now $x_i \leq d_i$ instead of $x_i \geq d_i$ as in the f-version.)

4.1 The Complexity of the Grouping Algorithm

In general, the membership problem in **Step 3** is as hard as the submodular function minimization itself. However, the special structure obtained through our construction of the subgroup will enable us to solve this problem much easily. The key is the following important observation,

Lemma 7 *In (26), only the side constraints and the constraint for the full set can be tight.*

Proof We prove this by induction. The case of two variables is obviously trivial. To illustrate the main idea, we first consider the case of three, then the general case. Now, suppose that subgroup g_j^i consists of two elements, k_1 and k_2. According to the **Step 2** of the grouping algorithm, we have,

$$f(G^{(i-1)}; g_1^i, \cdots, d_{j-1}^i) + d_{k_j} \leq f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, k_j), j = 1, 2 \tag{30}$$

$$f(G^{(i-1)}, g_1^i, \cdots, g_{j-1}^i, k_j, k_\ell) - f(G^{(i-1)}, g_1^i, \cdots, g_{j-1}^i, k_j) < d_{k_\ell}, j, \ell = 1, 2. \tag{31}$$

Then, adding variable k_3 that satisfies,

$$f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i) + d_{k_3} \leq f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, k_3),$$

and

$$f(G^{(i-1)}; g_1^i, \cdots, g_j^i) + d_{k_3} > f(G^{(i-1)}; g_1^i, \cdots, g_j^i, k_3).$$

First, we know that,

$$f(G^{(i-1)}; g_1^i, \cdots, d_{j-1}^i, k_3) + d_{k_j} \leq f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, k_3, k_j), j = 1, 2$$

can not be be valid for both $j = 1, 2$, since in this case, k_3 will form a subgroup of its own according to Step 2 of the grouping algorithm. From the fact that we have at least one j^*, such that,

$$f(G^{(i-1)}; g_1^i, \cdots, d_{j-1}^i, k_3) + d_{k_{j^*}} < f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, k_3, k_{j^*}),$$

it implies that $x_{k_3} \leq g(k_3)$ can not be tight, for otherwise the solution is infeasible. Now consider the subsets of $\{k_1, k_3\}$ and $\{k_2, k_3\}$, they can not be tight, either. Otherwise, suppose that $\{k_1, k_3\}$ is tight, then

$$x_{k_2} \leq g(k_1, k_2, k_3) - g(k_1, k_3) \leq g(k_1, k_2) - g(k_1) < d_2$$

which leads to infeasibility of the problem. Here the first inequality is due to submodularity and the second one is just (31). Similar arguments apply to $\{k_2, k_3\}$.

For the general n case, we can repeat the similar argument. Suppose that we have a k_n added into the subgroup g_i^j. Then again, due to the fact that $k - n$ is admitted into the subgroup, the following inequality will hold for some existing member of g_j^i, j^*,

$$f(G^{(i-1)}; g_1^i, \cdots, d_{j-1}^i, k_n) + d_{k_{j^*}} < f(G^{(i-1)}; g_1^i, \cdots, g_{j-1}^i, k_n, k_{j^*}),$$

which implies that $x_{k_n} \leq g(k_n)$ can not be tight. For all the subset contains k_n, but not the full set, the above submodularity argument can be employed again to show that the corresponding inequality can not tight. For those do not contain k_n, the induction hypothesis tells us that they can not be tight. Hence, the proposition follows. □

Lemma 8 *The complexity of* **Step 3** *is $O(LP(n))$. Where $LP(n)$ denote the complexity of solving a linear program with no more than n constraints and n variables.*

Proof By complementary slackness, in the dual of (26), that is,

$$\max \sum_A y^A g(A) - \sum_{k_i} d_{k_i} z_{k_i}, s.t. \sum_{A \ni k_1} y^A - z_{k_1} \leq 1, \sum_{A \ni k_i} y^A - z_{k_i} \leq 0, \forall k_i \neq k_1, \tag{32}$$

the variables that corresponds to constraints other than the side constraints and the one for the full set equal to zero. Therefore, to solve the dual problem, we only need to solve a

linear program with $\ell + 1$ variables and ℓ constraints. That gives us the complexity bound of $O(LP(n))$. In fact, to check whether the original problem is feasible, we simply need to rule out the unboundness of the dual problem. □

To characterize the computational effort of the entire grouping algorithm, consider the point where we started Step 1 of the algorithm, i.e., when the n-th element is added to the $m-1$ groups that have already been formed for the problem of $n-1$ elements. Note that throughout the algorithm, whenever an element moves, individually or in group, it always moves into a higher-priority (lower-indexed) group, or to a higher-priority within a group; and the very first subgroup is invalid, an entire group is eliminated. (Indeed, the number of groups can only decrease.) Hence, for an element originally in the k-th group, the number of moves can be no greater than total number of subgroups ahead, with each move involving a fixed number of operations associated with checking the inequalities in (21),(22),(23), and (24)), and solving an LP. Therefore, to finalize the grouping with the additional n-th element, the computational effort involved is $O(n^2 \times LP(n))$. Since we (inductively) bring in the n elements one at the time, the overall effort of the grouping algorithm is $O(n^3 \times LP(n))$.

4.2 Connections to the Queueing Scheduling Problem

Return to the queueing scheduling problem considered in §2.2), with the side constraints.

Theorem 9 Consider the control problem in (7), with additional side constraints, $x_i \geq d_i$, $i \in E$, and suppose the c_i's follow (10). Solve the LP in (9). Let $G_1, \ldots, G_m$ be the groups formulated by applying the grouping algorithm. Then, the optimal control u^* is a priority rule among the groups, with G_1 having the highest priority, and G_m the lowest priority; and within each group, u^* is a randomized rule that enforces the side constraints (on the non-lead variables).

For the last statement in the above theorem, reason as follows. With side constraints, the optimal solution x^* to the LP in (9) returned by the grouping algorithm is an interior point of the original polymatroid $\mathcal{P}(f)$ (i.e., without the side constraints). Following Caratheodory's theorem (refer to, e.g., Chvátal [5]), x^* can be expressed as a convex combination of no more than n vertices of $\mathcal{P}(f)$, each of which corresponds to a priority rule. (Note that the equality $x(E) = f(E)$ has reduced the degree of freedom by one; hence, n, instead of $n+1$.) In other words, x^* can be realized by a control that is a *randomization* of at most n priority rules, with the coefficients of the convex combination used as the probabilities for the randomization.

In terms of implementing the optimal control, one can do better than randomization. It is known (e.g., Federgruen and Groenevelt [13]) that any interior point of the performance space can be realized by a particular dynamic scheduling policy, due originally to Kleinrock [24, 25], in which the priority index of each job present in the system grows proportionately to the time it has spent waiting in queue, and the server always serves the job that has the highest index. This scheduling policy is completely specified by the proportionate coefficients associated with the jobs classes, which, in turn, are easily determined by the performance vector (provided it is at the interior of the performance space). In terms of practical implementation, there are several versions of this scheduling policy, refer to [16, 17].

5 Submodular Function Minimization

Suppose we want to minimize a submodular function $\psi(A)$, $A \subseteq E$. Following [7], we can transform this into an equivalent problem as follows. Define

$$d_i := \psi(E - \{i\}) - \psi(E), \qquad i \in E;$$

and write $d(A) = \sum_{i \in A} d_i$. Let

$$f(A) := \psi(A) + d(A) - \psi(\emptyset). \tag{33}$$

Then, minimizing

$$f(A) - d(A) = \psi(A) - \psi(\emptyset)$$

is equivalent to minimizing the original $\psi(A)$. Clearly, $f(\emptyset) = 0$, and f is submodular. To see that f is also increasing, consider $A \subset E$, and $i \notin A$. We have:

$$\begin{aligned} f(A + \{i\}) - f(A) &= \psi(A + \{i\}) + d(A) + d_i - \psi(A) - d(A) \\ &= \psi(A + \{i\}) - \psi(A) + \psi(E - \{i\}) - \psi(E) \\ &\geq 0, \end{aligned}$$

where the inequality follows from the submodularity of ψ.

Hence, the minimization of a submodular function $\psi(A)$ over $A \subseteq E$ is equivalent to the following problem:

$$\min_{A \subseteq E} \{f(A) - d(A)\}. \tag{34}$$

To solve this problem using the grouping algorithm of the last section, we still need (i) $d_j \geq 0$ for all $j \in E$, and (ii) $f(A) \geq d(A)$ for all $A \subseteq E$ (refer to Lemma 2).

Consider (i) first. If $d_j < 0$, for some j, then, for any set $A \subset E$, due to the submodularity of ψ, we have, for any negative d_j,

$$0 > d_j = \psi(E - \{j\}) - \psi(E) \geq \psi(A - \{j\}) - \psi(A), \tag{35}$$

implying $\psi(A-\{j\}) > \psi(A)$. That is, any minimizing set A will not contain j if $d_j < 0$. Hence, we can redefine the set E by first excluding such j's. This way, we can effectively assume $d_j \geq 0$ for all j.

The second requirement, $f(A) \geq d(A)$, following (33), is equivalent to $\psi(A) \geq \psi(\emptyset)$ for all A. To overcome this difficulty, we shall assume $\psi(A)$ is bounded from below: there exists an $M > 0$ such that $\psi(A) \geq -M$ for all $A \subseteq E$. Then, we can replace ψ by ψ' defined as follows:

$$\psi'(A) = \psi(A), \quad A \neq \emptyset; \qquad \psi'(\emptyset) = -M;$$

and $\psi'(A) \geq \psi'(\emptyset)$ for all $A \subseteq E$. We then solve the minimization problem in (34), but over $E - \emptyset$ (instead of E). Once the minimizing set A^* is identified, we can compare $\psi(A^*)$ with $\psi(\emptyset)$ to determine whether A^* or $\emptyset$ is optimal.

Theorem 10 Let G_1 be the first group formulated from applying the grouping algorithm (under the permutation $\pi = (1, ..., n)$). (Recall, x_1 is the lead variable in G_1.) Then, for any $B \neq G_1$, with $1 \in B \subseteq E$, B is inferior to G_1 in terms of yielding a larger objective value in (34).

Proof. First, suppose $B \subset G_1$. From Lemma 4, we have

$$f(B) \geq x(B), \qquad f(G_1) = x(A),$$

where x denotes the variable of the LP over $\mathcal{P}'(f)$ as in the last section. Hence,

$$f(G_1) - f(B) \leq x(G_1) - x(B) = d(G_1) - d(B),$$

where the equality follows from the fact that $1 \in B \subset G_1$; hence, $G_1 - B$ only contains non-lead variables.

Next, suppose $B \supset G_1$. Write $B = G_1 + A$, where $A \subseteq E - G_1$. We have, following Lemma **??**,

$$f(B) = f(A + G_1) \geq x(A) + x(G_1) = x(A) + f(G_1).$$

Hence,

$$f(B) - f(G_1) \geq x(A) \geq d(A) = d(B) - d(G_1).$$

Now, consider a general B. From submodularity, we have

$$f(G_1) + f(B) \geq f(G_1 \cup B) + f(G_1 \cap B).$$

Hence,

$$\begin{aligned}
& f(B) - d(B) \\
\geq\ & f(G_1 \cup B) + f(G_1 \cap B) - f(G_1) - d(B) \\
=\ & f(G_1 \cup B) - d(G_1 \cup B) + f(G_1 \cap B) - d(G_1 \cap B) - f(G_1) + d(G_1) \\
\geq\ & 2[f(G_1) - d(G_1)] - f(G_1) + d(G_1) \\
=\ & f(G_1) - d(G_1),
\end{aligned}$$

where the first equality follows from $d(\cdot)$ being additive, and the second inequality follows from the two cases already established above (regarding the subsets and supersets of G_1). □

Following Theorem 10, we can conclude that after the set $A = G_1$ is identified we do not have to consider the element 1 any more: any set that includes it must yield a larger objective value in (34) than G_1. To obtain the solution to problem(34), let us add another variable $\{0\}$ into the problem, define $d_0 = -1$ and $f(A \cup \{0\}) = f(A)$, this again defined a submodular function on $\hat{E} = E \cup \{0\}$. In the running of the algorithm, give 0 the highest priority, then it becomes the lead variable in G_1, hence the algorithm will produce,

$$\min_{A \subseteq \hat{E} - \{0\}} \{f(A) - d(A)\}.$$

However, from the definition, we know that 0 will never belongs to the minimum, hence, this is the optimum of (34).

Remark 1 Another way to view Theorem 10 is to consider the minimization problem in (34), with the restriction that $A \ni 1$, i.e.,

$$\min_{1 \in A \subseteq E} \{f(A) - d(A - \{1\})\}. \tag{36}$$

Note that we have dropped the $-d_1$ term in (36). The problem in (36) can be related to the LP over $\mathcal{P}'(f)$, with the following cost coefficients:

$$c_1 = 1, \quad c_2 = \cdots = c_n = 0.$$

Then, (36) becomes the same as the dual LP problem in (16), with $y_{G_1} = 1$, and $y_A = 0$ for all other $A = G^{(j)}$, $j = 2, ..., m$; $z_{k_j} = 1$, $k_j \in G_1^S$, and all other $z_i = 0$. Hence, the grouping algorithm will return $A = G_1$ as the optimal solution to (36).

This also suggests an alternative approach to the SFM in (34): for $i = 1, \ldots, n$, solve $\min_{i \in A \subseteq E}\{f(A) - d(A - \{i\})\}$, by applying the grouping algorithm to the LP over $\mathcal{P}'(f)$, with $c_i = 1$ and $c_j = 0$ for $j \neq i$. Each results in a subset $G_1 \ni i$. Clearly, only these n subsets need to be considered; hence, we can compare these in terms of their corresponding objective values in (34), and pick the the smallest one. □

Remark 2 The starting point of virtually all existing SFM algorithms in the literature (in particular, those overviewed in the introductory section) is to relate the problem in (34) to the following equivalent LP:

$$\max\{x(E) : \ x \in \mathcal{P}(f), \ x_i \leq d_i, \ i \in E\}, \tag{37}$$

where $\mathcal{P}(f)$ is the polymatroid in (1). (The equivalence between (34) and (37) is established in [10]; also refer to [27].) The above LP is then solved via finding augmenting paths in an associated graph. In this approach, the key step is to make sure that any improvement on the current solution y remains within the (base) polymatroid $\mathcal{B}(f)$, and this is accomplished via Carathéodory's representation of x (as a convex combination of the vertices of the polymatroid). As commented by McCormick in [27], "this is a rather brute-force way to verify that $y \in \mathcal{B}(f)$. However, 30 years of research have not yet produced a better idea."

In contrast, as explained in Remark 1, our approach is to solve the SFM in (34) via n subproblems such as the one in (36), which corresponds to the dual of an LP over $\mathcal{P}'(f)$. Note that $\mathcal{P}'(f)$ is different from the polytope in (37) — the inequalities in the side constraints are in the opposite direction. This subtle distinction turns out to be crucial: While the grouping algorithm solves the LP over $\mathcal{P}'(f)$, it is not clear how the algorithm can be adapted to solve the LP in (37). Recall, the intuition behind the grouping algorithm is that each job class associated with a non-lead variable, a non-contributor to maximizing the objective function, is moved into a higher-priority group just to meet the *lower* limit on its performance (throughput) requirement. Should the side constraints become upper limits, the structure of the solution would have to be completely different. □

Acknowledgements

We are much indebted to Shuzhong Zhang for his detailed and insightful comments on an earlier version of the paper. We also thank Jay Sethuraman and Li Zhang for useful discussions at various stages of this research.

References

[1] Bertsimas, D., The Achievable Region Method in the Optimal Control of Queueing Systems; Formulations, Bounds and Policies. *Queueing Systems,* **21** (1995), 337-389.

[2] Bertsimas, D. and Niño-Mora, J., Conservation Laws, Extended Polymatroids and Multi-Armed Bandit Problems: A Unified Approach to Indexable Systems. *Mathematics of Operations Research*, **21** (1996), 257-306.

[3] Bertsimas, D., Paschalidis, I.C. and Tsitsiklis, J.N., Branching Bandits and Klimov's Problem: Achievable Region and Side Constraints. *IEEE Trans. Aut. Control,* **40** (1995), 2063-2075.

[4] Bixby, R.E., Cunningham, W.H. and Topkis, D.M., The Poset of a Polymatroid Extreme Point. *Math. of Oper. Res.,* **10** (1985), 367-378.

[5] Chvátal, V., *Linear Programming.* W.H. Freeman, New York, 1983.

[6] Coffman, E. and Mitrani, I., A Characterization of Waiting Time Performance Realizable by Single Server Queues. *Operations Research,* **28** (1980), 810-821.

[7] Cunningham, W.H., On Submodular Function Minimization, *Combinatorica* **5**(3) 1985, 185-192

[8] Chen, H. and Yao, D.D., *Fundamentals of Queueing Networks: Performance, Asymptotics and Optimization.* Springer-Verlag, New York, 2001.

[9] Dunstan, F.D.J. and Welsh, D.J.A., A Greedy Algorithm for Solving a Certain Class of Linear Programmes. *Math. Programming,* **5** (1973), 338-353.

[10] Edmonds, J., Submodular Functions, Matroids and Certain Polyhedra. *Proc. Int. Conf. on Combinatorics (Calgary)*, Gordon and Breach, New York, 69-87, 1970.

[11] Federgruen, A. and Groenevelt, H., The Greedy Procedure for Resource Allocation Problems: Necessary and Sufficient Conditions for Optimality. *Operations Res.,* **34** (1986), 909-918.

[12] Federgruen, A. and Groenevelt, H., Characterization and Optimization of Achievable Performance in Queueing Systems. *Operations Res.,* **36** (1988), 733-741.

[13] Federgruen, A. and Groenevelt, H., M/G/c Queueing Systems with Multiple Customer Classes: Characterization and Control of Achievable Performance under Non-Preemptive Priority Rules. *Management Science,* **34** (1988), 1121-1138.

[14] Fleischer, L., Recent Progress in Submodular Function Minimization, *OPTIMA: Mathematical Programming Society Newsletter,* September 2000, no. 64, 1-11.

[15] Fleischer, L. and Iwata, S., A Push-Relabel Framework for Submodular Function Minimization and Applications to Parametric optimizations. To appear on *Discrete Applied Mathematics*, 2001.

[16] Fong, L.L. and Squillante, M.S., Time-Function Scheduling: A General Approach to Controllable Resource Management. IBM Research Report RC-20155, IBM Research Division, T.J. Watson Research Center, Yorktown Hts., New York, NY 10598, 1995.

[17] Franaszek, P.A. and Nelson, R.D., Properties of Delay Cost Scheduling in Timesharing Systems. IBM Research Report RC-13777, IBM Research Division, T.J. Watson Research Center, Yorktown Hts., New York, NY 10598, 1990.

[18] Fujishige, S., Lexicographically Optimal Base of a Polymatroid with respect to a Weight Vector. *Math. Ops. Res.,* **5** (1980), 186-196.

[19] Grötschel, M., Lovász, L and Schrijver, A., The Ellipsoid Method and Its Consequences in Combinatorial Optimization. *Combinatorica*, **1** (1981), 169-197.

[20] Grötschel, M., Lovász, L., and Schrijver, A., *Geometric Algorithms and Combinatorial Optimization*, second corrected edition. Springer-Verlag, Berlin, 1993.

[21] Iwata, S., A Fully Polynomial Algorithm for Submodular Function Minimizaton. *J. Comb. Theory, Ser. B*, **84** (2002), 203-212.

[22] Iwata, S., A Faster Scaling for Minimizing Submodular Functions. To appear in *The Procedings of IPCO 9,* (Cambridge, MA), 2002.

[23] Iwata, S., Fleischer, L. and Fujishige, S., A Strongly Polynomial-Time Algorithm for Minimizing Submodular Functions. Proceedings of the 32nd ACM Symposium on Theory of Computing, May, 2000. To appear in *Journal of the ACM.*

[24] Kleinrock, L., A Delay Dependent Queue Discipline. *Naval Research Logistics Quarterly,* **11** (1964), 329-341.

[25] Kleinrock, L., *Queueing Systems,* Vol. 2. Wiley, New York, 1976.

[26] Lu, Y., *Dynamic Scheduling of Stochastic Networks with Side Constraints.* Ph.D. Thesis, Columbia University, 1998.

[27] McCormick, T., Submodular Function Minimization. In: *Handbooks on Discrete Optimization*, *Preprint*, 2002.

[28] Ross, K.W. and Yao, D.D., Optimal Dynamic Scheduling in Jackson Networks. *IEEE Transactions on Automatic Control,* **34** (1989), 47-53.

[29] Ross, K.W. and Yao, D.D., Optimal Load Balancing and Scheduling in a Distributed Computer System. *Journal of the Association for Computing Machinery,* **38** (1991), 676-690.

[30] Shanthikumar J.G. and Yao D.D., Multiclass Queueing Systems: Polymatroid Structure and Optimal Scheduling Control. *Operation Research*, **40** (1992), Supplement 2, S293-299.

[31] Shrijver, A., A Combinatorial Algorithm Minimizing Submodular Functions in Polynomial Time. *J. Comb. Theory, Ser.B*, **80** (2000), 346-355.

[32] Welsh, D., *Matroid Theory*, (1976), Academic Press, London.

[33] Yao, D.D. and Zhang, L., Stochastic Scheduling and Polymatroid Optimization. In: *Lecture Notes in Applied Mathematics,* **33**, G. Ying and Q. Zhang (eds.), Springer-Verlag, 1997, 333-364.

[34] Zhang, L., *Reliability and Dynamic Scheduling in Stochastic Networks.* PhD. Thesis, Columbia University, 1997.